Partial Higher-Dimensional Automata

Uli Fahrenberg, Axel Legay

Partial Higher-Dimensional Automata

Uli Fahrenberg and Axel Legay

INRIA/IRISA, Campus de Beaulieu, 35042 Rennes CEDEX, France

Abstract

We propose a generalization of higher-dimensional automata, partial HDA. Unlike HDA, and also extending event structures and Petri nets, partial HDA can model phenomena such as priorities or the disabling of an event by another event. Using open maps and unfoldings, we introduce a natural notion of (higher-dimensional) bisimilarity for partial HDA and relate it to history-preserving bisimilarity and split bisimilarity. Higher-dimensional bisimilarity has a game characterization and is decidable in polynomial time.

1998 ACM Subject Classification F.1.1 Models of Computation

1 Introduction

Higher-dimensional automata (HDA) is a formalism for modeling and reasoning about behavior of concurrent systems. Like Petri nets [22], event structures [20], configuration structures [32], asynchronous transition systems [1, 27] and other similar formalisms, it is *non-interleaving* in the sense that it differentiates between concurrent and interleaving events; using CCS notation [19], $a|b \neq a.b + b.a$.

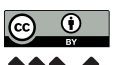
Introduced by Pratt [23] and van Glabbeek [29] in 1991 for the purpose of a *geometric* interpretation to the theory of concurrency, it has since been shown by van Glabbeek [30] that HDA provide a generalization (up to *history-preserving bisimilarity*) to "the main models of concurrency proposed in the literature" [30], including the ones mentioned above. Hence HDA are useful as a tool for comparing and relating different models, and also as a modeling formalism by themselves.

HDA are geometric in the sense that they are similar to the *simplicial complexes* used in algebraic topology, and research on HDA has drawn on tools and methods from geometry and topology such as homotopy [5, 8–10], homology [14], and model categories [11, 12], see also the surveys [13, 15].

Motivated by some examples of concurrent systems which *cannot* be modeled by HDA, we propose here an extension of the formalism, called *partial* or *incomplete* HDA. Intuitively, these are HDA in which some parts may be missing; transitions which do not have an end state, squares which miss parts of their boundary, etc. We will show that these can be used to model phenomena such as priorities and the disabling of events by other events.

We show that partial HDA admit a natural notion of bisimilarity, defined categorically through open maps in the spirit of Joyal, Nielsen and Winskel [17, 35]. (We have included a background section to introduce and motivate the categorical setting.) This opens up for using coinductive techniques for (partial) HDA. We also give a game characterization of this *hd-bisimilarity* and show that is decidable for finite partial HDA.

We then define unfoldings of partial HDA into higher-dimensional trees, which are given as the equivalence classes of computation paths under a certain notion of homotopy of computations, rather similarly to universal coverings in algebraic topology. These unfoldings are used to express hd-bisimilarity as an equivalence relation on homotopy classes of computations and, ultimately, directly on computations. This allows us to compare hd-bisimilarity to other common notions of equivalences for concurrent models, such as split bisimilarity [30],



ST-bisimilarity [33] and history-preserving bisimilarity [25, 31]. We show that hd-bisimilarity is strictly weaker than history-preserving bisimilarity, but not weaker than split bisimilarity.

We start the paper by giving some categorical background for our developments in Section 2, with the purpose of introducing just enough category theory so that the rest of the paper, except perhaps for the last section, can be understood also by readers without a categorical inclination. Section 3 then introduces partial HDA and shows important examples of systems which can be modeled only as partial HDA. In Section 4 we then introduce our notion of hd-bisimilarity through open maps in the category of partial HDA. We give an elementary characterization of hd-bisimilarity in Theorem 9 and a characterization using games in Theorem 12.

Section 5, introducing homotopy of computations and unfoldings of partial HDA, is the technical core of the paper. Its central result is Corollary 16, that partial HDA are hd-bisimilar iff their unfoldings are so. This result is used for comparing hd-bisimilarity with other equivalences for concurrent models in Section 6, showing in Theorem 18 our main result that hd-bisimilarity is strictly weaker than history-preserving bisimilarity but not weaker than split bisimilarity.

For (total) HDA, the categorical setting on which our work is built was first introduced in [4, 5]. It has the advantage of a close analogy to the simplicial and cubical sets used in algebraic topology [3, 16, 26]. Later we have connected this work to history-preserving bisimilarity in [6], see also [7] for some corrections. Note that the version of hd-bisimilarity introduced in our earlier work [6, 7] for HDA is different from the one we define here; indeed the earlier variant is incomparable with history-preserving bisimilarity. This is essentially because HDA are required to have all boundaries and is avoided by passing to partial HDA.

Acknowledgments

The authors wish to thank Cristian Prisacariu and Rob van Glabbeek for enlightening discussions on the subject of this paper, and the organizers of SMC 2014 in Lyon for providing a forum for these discussions.

2 Categorical Background

To warm up, we review some of the work in [17, 35] on the category of transition systems and open maps for bisimulations, modified slightly to suit our purposes. This categorical setting is useful for us, because it allows to state properties in an abstract generality which allows for immediate generalization to other settings. More specifically, the work of Joyal, Winskel and Nielsen in [17, 35] and other papers has been influential because through the categorical setting, properties can be stated and proven across formalisms and easily be transferred from one formalism to another. This has exposed some very useful similarities between formalisms which look very different, for example transition systems, Petri nets, and event structures. Hence category theory is useful here as an ordering principle.

2.1 Digraphs

A *digraph* $X = (X_1, X_0)$ consists of two sets X_1, X_0, of edges and vertices, together with *face maps* $\delta^0, \delta^1 : X_1 \to X_0$ assigning start and end vertices to every edge. Note that we allow loops and multiple edges in our digraphs.

A *morphism* of digraphs $f : X \to Y$ consists of two mappings $f_1 : X_1 \to Y_1$, $f_0 : X_0 \to Y_0$ which commute with the face maps, *i.e.* such that $f_0(\delta^0 a) = \delta^0 f_1(a)$ and $f_0(\delta^1 a) = \delta^1 f_1(a)$

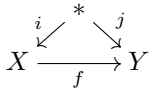
for every edge $a \in X_1$. Hence morphisms are standard digraph homomorphisms.

Digraphs and their morphisms form a *category*, in that composition of morphisms is associative and every digraph X has an identity morphism id^X given by $\mathrm{id}^X_1(a) = a$ and $\mathrm{id}^X_0(x) = x$. We will denote this category by Dgr.

2.2 Transition Systems

A *transition system* (X, i_0) is a digraph X with a specified initial vertex (state) $i_0 \in X_0$. This is the same as specifying a mapping $i_0 : \{0\} \to X_0$ from a one-point set into the vertices, which can be extended (uniquely) to a morphism $i : * \to X$ from the *one-point digraph* (without edges). We have hence transferred an *internal* object, an element $i_0 \in X_0$, to an *external* setting, a morphism $i : * \to X$. This process of *externalization* is very important in applications of category theory, as it allows to transfer properties internal to objects or morphisms (here the very simple property of having a specified initial element) to an external setting which only uses objects and morphisms as-is.

Morphisms of transition systems are required to respect the initial states, *i.e.* if $f : (X, i) \to (Y, j)$ is such a morphism, then we must have $f(i) = j$. This is the same as saying that the category of transition systems is the *comma category* (or slice) of digraphs under the object $*$: objects are digraph morphisms $* \to X$ and morphisms are digraph morphisms $f : X \to Y$ for which the diagram

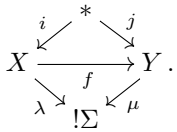


commutes. We denote this comma category by $* \downarrow \mathsf{Dgr}$.

2.3 Labeled Transition Systems

A *labeled transition system* (LTS) (X, i, λ_1), over some alphabet Σ, is a transition system $i : * \to X$ together with an edge labeling $\lambda_1 : X_1 \to \Sigma$. We externalize the edge labeling: Let $!\Sigma = (\{0\}, \Sigma)$ be the one-point digraph with edge set Σ (*i.e.* the digraph with one vertex 0 and $\delta^0\alpha = \delta^1\alpha = 0$ for every $\alpha \in \Sigma$), then any mapping $\lambda_1 : X_1 \to \Sigma$ can be extended (uniquely) to a digraph morphism $\lambda : X \to !\Sigma$. A LTS is, then, a system of digraph morphisms $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$: $i : * \to X$ specifies the initial state, and $\lambda : X \to !\Sigma$ associates labels to edges.

Morphisms of LTS $(X, i, \lambda_1) \to (Y, j, \mu_1)$ are digraph morphisms $f : X \to Y$ which respect the initial state and the labeling: for every $a \in X_1$, $\mu_1(f_1(a)) = \lambda_1(a)$. (For simplicity we only consider such label-preserving morphisms here; this is all we will need later.) This is the same as saying that the category of LTS is the double comma category $* \downarrow \mathsf{Dgr} \downarrow !\Sigma$ of digraphs between $*$ and $!\Sigma$: objects are structures $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$ and morphisms are commutative diagrams



Posing initial states and labels as a double comma category has the advantage that many constructions can be simply defined on the base category (here: digraphs; below: partial

precubical sets) and then lifted to the double comma category. We will exploit this below to do most of our work in the unlabeled category (of partial HDA) and only in the last section lift it to the labeled setting.

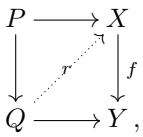
2.4 Open Maps and Bisimilarity

A digraph morphism $f : X \to Y$ is called an *open map* if it holds that for all $x \in X_0$ and $b \in Y_1$ with $\delta^0 b = f_0(x)$, there exists $a \in X_1$ with $\delta^0 a = x$ such that $b = f_1(a)$. Hence any edge b which starts in $f_0(x)$ can be lifted (not necessarily uniquely) to an edge a, emanating from x, for which $b = f_1(a)$.

One of the contributions of [17] is the lift of the above open maps to the usual *relational* setting of bisimulation [19, 21]: by a theorem of [17], two LTS X, Y are *bisimilar* iff there exists an LTS Z and a *span of open maps* $X \leftarrow Z \to Y$.

2.5 Path Objects

To externalize the property of being open, one defines a category of *path objects* (or *computations*). A path object is a transition system $(\{x_0, \dots, x_n\}, \{(x_0, x_1), \dots, (x_{n-1}, x_n)\}, x_0)$, for $n \geq 0$, *i.e.* a path in the graph-theoretical sense, with distinct states $x_0, \dots, x_n$ and transitions from x_i to x_{i+1} for all $i = 0, \dots, n-1$. Morphisms of path objects are inclusions of shorter paths into longer ones (hence path objects form a *full* subcategory of transition systems). It can then be shown that a transition system morphism $f : X \to Y$ is open iff there is a morphism (a *lift*) $r : Q \to X$ in any diagram of the form



where P and Q are path objects.

We have established that bisimulation of LTS can be posed in an entirely external categorical setting, where two LTS are bisimilar iff here is a span of morphisms which have a special property of being open which is defined through a (right) lifting property with respect to a subcategory of paths. This will be a guidance for our developments in later sections.

3 Partial Higher-Dimensional Automata

Higher-dimensional automata [23, 29] generalize transition systems in the sense that they allow for higher-dimensional transitions: they admit states and transitions, but also two-dimensional (squares) and three-dimensional (cubes) transitions, *etc.* *Partial* HDA, as presented in this paper, are a further generalization in which some transitions may not have start or end states, or squares may not have some of their start or end transitions, *etc.* As in the preceding section, we define partial HDA using a comma category construction.

3.1 HDA

We start by recalling HDA. A *precubical set* is a graded set $X = \{X_n\}_{n \in \mathbb{N}}$ together with mappings $\delta^\nu_{k(n)} : X_n \to X_{n-1}$, $k \in \{1, \dots, n\}$, $\nu \in \{0, 1\}$, satisfying the *precubical identity*

$$\delta^\nu_k \delta^\mu_\ell = \delta^\mu_{\ell-1} \delta^\nu_k \qquad (k < \ell)\,.$$

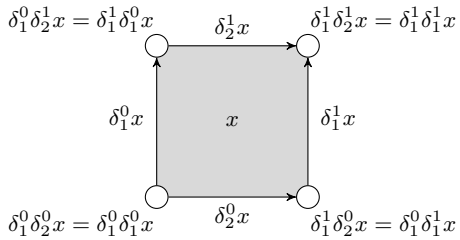


Figure 1 A 2-cube x with its four faces $\delta^0_1 x$, $\delta^1_1 x$, $\delta^0_2 x$, $\delta^1_2 x$ and four corners.

The mappings $\delta^\nu_{k(n)}$ are called *face maps* (note that we will omit the extra index (n)), and elements of X_n are called *n-cubes*. Faces $\delta^0_k x$ of an element $x \in X$ are to be thought of as *start faces*, $\delta^1_k x$ as *end faces*. The precubical identity expresses the fact that $(n-1)$-faces of an n-cube meet in common $(n-2)$-faces; see Fig. 1 for an example. Note how this generalizes digraphs to arbitrary dimensions: a precubical set includes vertices and edges, and some squares of edges may be filled in, some cubes of squares may be filled in, *etc.*

Similarly to digraph morphisms, morphisms $f : X \to Y$ of precubical sets are graded functions $f = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ which commute with the face maps: $\delta^\nu_k \circ f_n = f_{n-1} \circ \delta^\nu_k$ for all $n \in \mathbb{N}$, $k \in \{1,\dots,n\}$, $\nu \in \{0,1\}$. This defines a category of precubical sets.

The category of *HDA* is then the comma category of precubical sets under the one-point precubical set $*$ with one 0-cube and no other n-cubes. Hence a one-dimensional HDA is a transition system; indeed, the category of transition systems [35] is isomorphic to the full subcategory of one-dimensional HDA.

3.2 Partial HDA

The following example exposes a simple system which cannot be modeled as HDA; this motivates the introduction of partial HDA below.

▶ **Example 1.** Let a and b be two independent events (which hence may run concurrently) with the constraint that b cannot start before a and has to finish before a can finish. Hence b can only run "inside" a; by way of motivation, a could be a supervisor process which provides resources for b. (Hence this is an example of the disabling of an event by another event.)

Note that this system cannot be modeled as an event structure. We can represent it as an *ST-structure* as introduced in [24], which is comprised of configurations (S,T) of *started* (S) and *terminated* (T) events (hence always $T \subseteq S$):

$$(\emptyset,\emptyset) \xrightarrow[s]{a} (\{a\},\emptyset) \xrightarrow[s]{b} (\{a,b\},\emptyset) \xrightarrow[t]{b} (\{a,b\},\{b\}) \xrightarrow[t]{a} (\{a,b\},\{a,b\})$$

When trying to model this example as a HDA, *cf.* Fig. 2 below, we see that existence of the 2-cube corresponding to the configuration $(\{a,b\},\emptyset)$ forces us to introduce all its boundaries into the model, *i.e.* not only the configurations $(\{a\},\emptyset)$ and $(\{a,b\},\{b\})$ as above, but also $(\{b\},\emptyset)$ and $(\{a,b\},\{a\})$. Thus we lose the property that b can only run inside a.

We hence define a *partial precubical set* (PPS) to be a graded set $X = \{X_n\}_{n\in\mathbb{N}}$ together with *partial* mappings $\delta^\nu_k : X_n \hookrightarrow X_{n-1}$, $k \in \{1,\dots,n\}$, $\nu \in \{0,1\}$, satisfying the precubical identity

$$\delta^\nu_k \delta^\mu_\ell = \delta^\mu_{\ell-1}\delta^\nu_k \qquad (k < \ell) \tag{1}$$

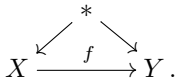
whenever all the involved mappings are defined. We will always assume the sets X_n to be disjoint. For an n-cube $x \in X_n$, we denote by $\dim x = n$ its *dimension*.

Morphisms $f : X \to Y$ of PPS are graded *total* functions $f = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ which commute with the face maps: $\delta^\nu_k \circ f_n = f_{n-1} \circ \delta^\nu_k$ for all $n \in \mathbb{N}$, $k \in \{1, \dots, n\}$, $\nu \in \{0, 1\}$ whenever the involved face maps are defined. This defines a category PPS of partial precubical sets and morphisms.

Products of PPS are given point-wise: for PPS X, Y, $X \times Y = Z$ with $Z_n = X_n \times Y_n$ and face maps defined by $\delta^\nu_k(x, y) = (\delta^\nu_k x, \delta^\nu_k y)$ iff both individual faces are defined. (This is the categorical product.) Also *subsets* are given point-wise: for $X, Y \in \mathsf{PPS}$, $Y \subseteq X$ iff $Y_n \subseteq X_n$ for all $n \in \mathbb{N}$.

A *pointed* PPS is a PPS X with a specified 0-cube $i \in X_0$, and a pointed morphism is one which respects the point. This defines a category which is isomorphic to the comma category $* \downarrow \mathsf{PPS}$, where $* \in \mathsf{PPS}$ is the precubical set with one 0-cube and no other n-cubes.

▶ **Definition 2.** The category of *partial higher-dimensional automata* (PHDA) is the comma category $\mathsf{PHDA} = * \downarrow \mathsf{PPS}$, with objects pointed PPS and morphisms commutative diagrams



Intuitively, 0-cubes $x \in X_0$ are to be thought of as *states*, 1-cubes are *transitions*, and n-cubes for $n \geq 2$ model concurrent executions of n events. Note that a one-dimensional PHDA is a transition system in which transitions do not necessarily have start or end states. This may be useful for modeling deadlocks, even though we are not aware of any work in which this is done.

3.3 Labeled Partial HDA

For *labeling* PHDA, we let $\Sigma = \{a_1, a_2, \dots\}$ be a finite or infinite set of events. We construct a precubical set $!\Sigma = \{!\Sigma_n\}$ by letting $!\Sigma_n = \{(a_{i_1}, \dots, a_{i_n}) \mid i_k \leq i_{k+1} \text{ for all } k = 1, \dots, n-1\}$ with face maps defined by $\delta^\nu_k(a_{i_1}, \dots, a_{i_n}) = (a_{i_1}, \dots, a_{i_{k-1}}, a_{i_{k+1}}, \dots, a_{i_n})$. Note that $!\Sigma$ is a *torus*: start and end faces of any n-cube agree, hence all n-cubes are loops.

▶ **Definition 3.** The category of *labeled PHDA* over Σ is the double comma category $\mathsf{LHDA} = * \downarrow \mathsf{PPS} \downarrow !\Sigma$.

Note that this labels opposite transitions with the same event, *i.e.* $\lambda\delta^0_1 z = \lambda\delta^1_1 z$ and $\lambda\delta^0_2 z = \lambda\delta^1_2 z$, for every $z \in X_2$, whenever these boundaries exist in X_1. This conveys the intuition that opposite boundaries of a square execute the same event, connected by possible concurrent execution of another event. We develop most of the material in this paper for unlabeled PHDA and transport it to the labeled setting in Section 6.

▶ **Example 4.** We can now expose a labeled PHDA model for the system of Example 1. Let $X \in \mathsf{PHDA}$ be such that $X_n = \emptyset$ for $n \geq 3$, $X_2 = \{z\}$, $X_1 = \{y_1, y_2\}$ and $X_0 = \{x_0, x_2\}$, with face maps $\delta^0_2 z = y_1$, $\delta^1_2 z = y_2$, $\delta^0_1 y_1 = x_0$, $\delta^1_1 y_2 = x_2$ (and all others undefined), initial state x_0 and labeling $\lambda(y_1) = \lambda(y_2) = a$, $\lambda(z) = ab$, see Fig. 2. The computational interpretation of X is that b can only start while a is executing, and a can only finish once b is done.

▶ **Example 5.** For a slightly more involved example, let again a and b be independent events, but this time so that a is executed in a loop; once b has started, a cannot be started anymore; and b can only finish when a is not running (hence b has priority over a). By way

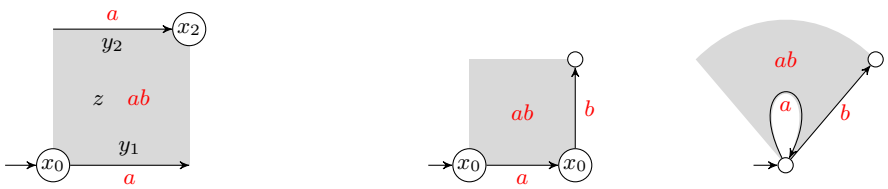


Figure 2 Labeled PHDA of Examples 4 and 5. The gray area signifies a 2-cube; labels are indicated in red.

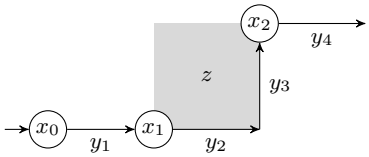


Figure 3 The two-dimensional path object $(x_0, y_1, x_1, y_2, z, y_3, x_2, y_4)$. Its computational interpretation is that y_1 is executed first; after it finishes, y_2 is started, and while y_2 is running, y_3 starts to execute. After this, y_2 finishes, then y_3 finishes, and then execution of y_4 is started. Note that the computation is partial, as y_4 does not finish.

of motivation, b could be a "shutdown" process which waits for other processes to terminate but does not allow new ones to start. As a labeled PHDA, this can be modeled as in Fig. 2. Note that this PHDA contains a cycle; the two copies of x_0 on the left indicate that they are to be identified, as can be seen on the right.

4 Higher-Dimensional Bisimilarity

Following the procedure outlined in Section 2, we now introduce path objects, define open maps as these morphisms which have the right-lifting property with respect to the path category, and use this to define bisimilarity. This is similar to what we did in [6], but because we are working with *partial* HDA, things are closer to the computational intuition.

4.1 Path Objects

We say that a PPS X is a *path object* if its n-cubes can be sorted into a (necessarily unique) sequence $(x_1, \dots, x_m)$ such that $x_i \neq x_j$ for $i \neq j$, for each $j = 1, \dots, m-1$, there is $k \in \mathbb{N}$ for which $x_j = \delta^0_k x_{j+1}$ or $x_{j+1} = \delta^1_k x_j$, and no other relations exist between the x_i. Hence a path object is a sequence of cubes which are connected so that either x_{j+1} is an extension of x_j, signifying the start of a new event, or x_{j+1} is an end face of x_j, signifying the end of an event, see Fig. 3 for an example.

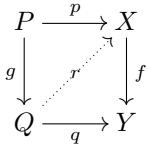
A *pointed* path object $i : * \to X$ consists of a path object X and the mapping i which includes the point as x_1 (hence $x_1 \in X_0$). Intuitively, path objects are models of PHDA computations, just as paths are models of transition system computations (Section 2). Pointed path objects are computations from an initial state.

If X and Y are path objects with representations $(x_1, \dots, x_m)$, $(y_1, \dots, y_p)$, then a morphism $f : X \to Y$ is called a *cube path extension* if $x_j = y_j$ for all $j = 1, \dots, m$ (hence $m \leq p$). This models the extension of one computation by zero or more steps, in analogy to extensions of paths in Section 2.

▶ **Definition 6.** The category HDP of *higher-dimensional paths* is the subcategory of PHDA which as objects has pointed path objects and whose morphisms are generated by pointed cube path extensions and isomorphisms.

4.2 Open Maps and Hd-bisimilarity

▶ **Definition 7.** A pointed morphism $f : X \to Y$ in PHDA is an *open map* if it has the right lifting property with respect to HDP, *i.e.* if it is the case that there is a lift r in any commutative diagram as below, for morphisms $g : P \to Q \in \mathsf{HDP}$, $p : P \to X, q : Q \to Y \in$ PHDA:



Note how this is entirely analogous to what we did in Section 2. Stating concepts in a categorical way has allowed us to transport them from transition systems to PHDA.

▶ **Definition 8.** PHDA X, Y are *hd-bisimilar* if there is $Z \in \mathsf{PHDA}$ and a span of open maps $X \leftarrow Z \rightarrow Y$ in PHDA.

A relational formulation of this is as follows:

▶ **Theorem 9.** *PHDA $i : * \to X$, $j : * \to Y$ are hd-bisimilar iff there exists a PPS $R \subseteq X \times Y$ for which $(i, j) \in R$, and such that for all $(x_1, y_1) \in R$,*

1. *for any $x_2 \in X$ for which $x_1 = \delta^0_k x_2$ for some k, there exists $y_2 \in Y$ for which $y_1 = \delta^0_k y_2$ and $(x_2, y_2) \in R$,*
2. *for any $x_2 \in X$ for which $x_2 = \delta^1_k x_1$ for some k, there exists $y_2 \in Y$ for which $y_2 = \delta^1_k y_1$ and $(x_2, y_2) \in R$,*
3. *for any $y_2 \in Y$ for which $y_1 = \delta^0_k y_2$ for some k, there exists $x_2 \in X$ for which $x_1 = \delta^0_k x_2$ and $(x_2, y_2) \in R$,*
4. *for any $y_2 \in Y$ for which $y_2 = \delta^1_k y_1$ for some k, there exists $x_2 \in X$ for which $x_2 = \delta^1_k x_1$ and $(x_2, y_2) \in R$.*

Proof. For the forward implication, let $X \xleftarrow{f} Z \xrightarrow{g} Y$ be a span of open maps and define $R = \{(x, y) \in X \times Y \mid \exists z \in Z : x = f(z), y = g(z)\}$. Then $(i, j) \in R$ because f and g are pointed morphisms, properties (1) and (3) hold because f and g are PPS morphisms, and properties (2) and (4) hold because f and g are open. For the backwards implication, let $\pi_X : R \to X$, $\pi_Y : R \to Y$ be the projections; these are easily shown to be open maps. ◀

▶ **Corollary 10.** *For finite PHDA, hd-bisimilarity is decidable in polynomial time.*

Proof. The condition in Theorem 9 immediately gives rise to a fixed-point algorithm similar to the one used to decide standard bisimilarity, *cf.* [18, 19]. ◀

▶ **Example 11.** The two (total) labeled HDA in Fig. 4 are hd-bisimilar, as witnessed by the following PPS $R \subseteq X \times X'$:

$$R_0 = \{(x_0, x'_0), (x_1, x'_1), (x_2, x'_2), (x_3, x'_4), (x_4, x'_4)\}$$
$$R_1 = \{(y_1, y'_1), (y_2, y'_2), (y_3, y'_4), (y_4, y'_4), (y_5, y'_5)\}$$
$$R_2 = \{(z, z')\}$$

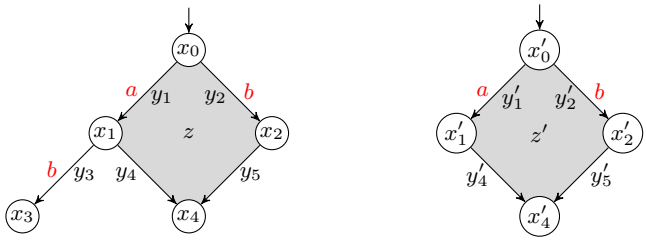


Figure 4 Two HDA pertaining to Example 11.

4.3 Hd-bisimulation Games

We can also expose a game characterization of hd-bisimilarity, similar to the notion of bisimulation game for interleaving bisimilarity [28]. The game is played by two players, *Spoiler* and *Duplicator*, and a configuration of the game is a pair (x, y) of n-cubes $x \in X$, $y \in Y$ of equal dimension. The initial configuration is (i, j).

At each round of the game, from a configuration (x_1, y_1), the spoiler chooses to play one of four moves: either

1. to choose $x_2 \in X$ with $x_1 = \delta^0_k x_2$ for some k,
2. to choose $x_2 \in X$ with $x_2 = \delta^1_k x_1$ for some k,
3. to choose $y_2 \in Y$ with $y_1 = \delta^0_k y_2$ for some k, or
4. to choose $y_2 \in Y$ with $y_2 = \delta^1_k y_1$ for some k.

Depending on the type of move of the spoiler, the duplicator now has to answer by, respectively,

1. choosing $y_2 \in Y$ with $y_1 = \delta^0_k y_2$,
2. choosing $y_2 \in Y$ with $y_2 = \delta^1_k y_1$,
3. choosing $x_2 \in X$ with $x_1 = \delta^0_k x_2$, or
4. choosing $x_2 \in X$ with $x_2 = \delta^1_k x_1$,

and the game continues from the configuration (x_2, y_2).

The spoiler wins the game if the duplicator gets stuck, *i.e.* if the game finishes because duplicator has no answer to a move of the spoiler. Otherwise (if the game is infinite, or if it finishes because the spoiler has no move) the duplicator has won. The proof of the following theorem is similar to the one for the game characterization of interleaving bisimilarity [28].

▶ **Theorem 12.** *PHDA X and Y are hd-bisimilar iff the duplicator has a winning strategy in the hd-bisimulation game between X and Y.*

5 Homotopy and Unfoldings

Most other common notions of equivalences for concurrent systems, such as (hereditary) history-preserving bisimilarity, ST-bisimilarity or split bisimilarity, are defined on computations rather than structurally (see [30]; we will define them formally below). Hence to compare our notion of hd-bisimilarity to these other equivalences, we need to lift it to a relation on computations. The vehicle for doing so is the *unfolding* of a PHDA, similar to the *universal covering space* in algebraic topology.

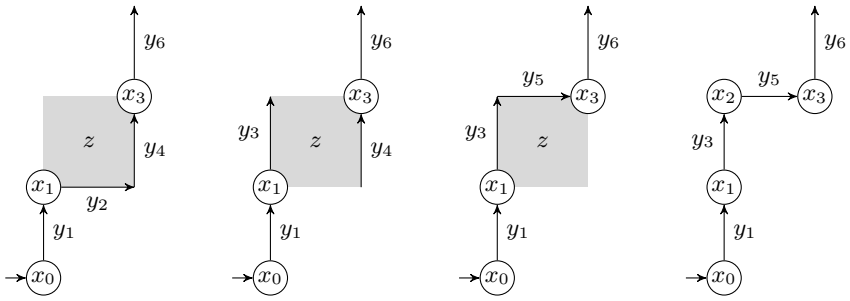


Figure 5 The cube path homotopy $(x_0, y_1, x_1, y_2, z, y_4, x_3, y_6) \sim (x_0, y_1, x_1, y_3, z, y_4, x_3, y_6) \sim (x_0, y_1, x_1, y_3, z, y_5, x_3, y_6) \sim (x_0, y_1, x_1, y_3, x_2, y_5, x_3, y_6)$.

5.1 Computations

We have already introduced path objects above, which embody the intuition behind PHDA computations. Using these to define computations *within* a given PHDA, we say that a *cube path* in a PPS X is a morphism $P \to X$ from a path object P. In elementary terms, this is a sequence $(x_1, \ldots, x_m)$ of elements of X such that for each $j = 1, \ldots, m-1$, there is $k \in \mathbb{N}$ for which $x_j = \delta^0_k x_{j+1}$ (start of a new part of a computation) or $x_{j+1} = \delta^1_k x_j$ (end of a computation part).

Note that cube paths, contrary to path objects, may have loops and self-intersections (conforming to the intuition that they be computations in a PHDA). As an example, the PHDA in Fig. 2 is not itself a path object, but any finite sequence of a-labeled transitions is a cube path within it, as is any finite sequence of a-labeled transitions followed by a b-labeled transition.

A *pointed* cube path in a PHDA $* \to X$ is a pointed morphism from a pointed path object. We will say that a cube path $(x_1, \ldots, x_m)$ is *from* x_1 *to* x_m, and that an n-cube $x \in X$ in a PHDA X is *reachable* if there is a pointed cube path to x in X.

5.2 Homotopy of Computations

We define an equivalence relation on cube paths which formalizes the intuition of when two concurrent computations are the same. We say that cube paths $\rho = (x_1, \ldots, x_m)$, $\sigma = (y_1, \ldots, y_m)$ are *p-adjacent*, and write $\rho \overset{p}{\sim} \sigma$, for $p \in \{2, \ldots, m-1\}$, if $x_p \neq y_p$ and $x_j = y_j$ for $j \neq p$, and one of the following conditions is satisfied:

- $x_{p-1} = \delta^0_k x_p$, $x_p = \delta^0_\ell x_{p+1}$, $y_{p-1} = \delta^0_{\ell-1} y_p$, and $y_p = \delta^0_k y_{p+1}$ for some $k < \ell$, or vice versa,
- $x_p = \delta^1_k x_{p-1}$, $x_{p+1} = \delta^1_\ell x_p$, $y_p = \delta^1_{\ell-1} y_{p-1}$, and $y_{p+1} = \delta^1_k y_p$ for some $k < \ell$, or vice versa,
- $x_p = \delta^0_k \delta^1_\ell y_p$, $y_{p-1} = \delta^0_k y_p$, and $y_{p+1} = \delta^1_\ell y_p$ for some $k < \ell$, or vice versa, or
- $x_p = \delta^1_k \delta^0_\ell y_p$, $y_{p-1} = \delta^0_\ell y_p$, and $y_{p+1} = \delta^1_k y_p$ for some $k < \ell$, or vice versa.

The intuition of adjacency is rather simple, even though the combinatorics may look complicated; see Fig. 5 for an example. Note that adjacencies come in two basic "flavors": the first two above in which the dimensions of x_p and y_p are the same, and the last two in which they differ by 2.

We say that two cube paths are adjacent if they are p-adjacent for some p, and *homotopy* of cube paths is defined to be the reflexive, transitive closure of the adjacency relation. We denote homotopy of cube paths using the symbol $\sim$, and the homotopy class of a cube path $(x_1, \ldots, x_m)$ is denoted $[x_1, \ldots, x_m]$.

5.3 Unfoldings

We will unfold PHDA into *higher-dimensional trees*, which are PHDA X for which it holds that there is precisely one homotopy class of cube paths to any n-cube $x \in X$. The full subcategory of PHDA spanned by the higher-dimensional trees is denoted HDT. Note that any path object is a higher-dimensional tree.

▶ **Definition 13.** The *unfolding* of a PHDA $i : * \to X$ consists of a PHDA $\tilde{i} : * \to \tilde{X}$ and a pointed *projection* morphism $\pi_X : \tilde{X} \to X$, which are defined as follows:

- $\tilde{X}_n = \{[x_1, \dots, x_m] \mid (x_1, \dots, x_m)$ pointed cube path in $X, x_m \in X_n\}$; $\tilde{i} = [i]$
- $\tilde{\delta}^0_k[x_1, \dots, x_m] = \{(y_1, \dots, y_p) \mid y_p = \delta^0_k x_m, (y_1, \dots, y_p, x_m) \sim (x_1, \dots, x_m)\}$ provided this set is non-empty; otherwise undefined
- $\tilde{\delta}^1_k[x_1, \dots, x_m] = [x_1, \dots, x_m, \delta^1_k x_m]$ if $\delta^1_k x_m$ exists; otherwise undefined
- $\pi_X[x_1, \dots, x_m] = x_m$

▶ **Theorem 14.** *The unfolding $(\tilde{X}, \pi_X)$ of a PHDA X is well-defined, and $\tilde{X}$ is a higher-dimensional tree. If X itself is a higher-dimensional tree, then the projection $\pi_X : \tilde{X} \to X$ is an isomorphism.*

Proof sketch. Note the complete analogy to the construction of universal covering spaces in algebraic topology: $\tilde{X}$ consists of homotopy classes of (cube) paths, and the projection maps a path to its end point (cube). The proof is similar to the one we gave for (total) HDA in [6], but with the important difference that a certain ("fan-shaped") normal form for cube paths, which we used in [6], is not available for partial HDA. We present a sketch of the proof here; the full proof is in appendix.

To see that $\tilde{X}$ is well-defined, we need to show that the face maps $\tilde{\delta}^0_k$ and $\tilde{\delta}^1_k$ are independent of the representative in the homotopy class. For $\tilde{\delta}^1_k$ this is trivial, but for $\tilde{\delta}^0_k$ it requires more work. We also need to prove that the precubical identity $\tilde{\delta}^\nu_k \tilde{\delta}^\mu_\ell = \tilde{\delta}^\mu_{\ell-1} \tilde{\delta}^\nu_k$ is satisfied whenever the faces exist; this is again trivial for $\nu = \mu = 1$ and more complicated for the other cases.

The projection $\pi : \tilde{X} \to X$ is clearly well-defined, as homotopic cube paths have identical end points. To see that it is a PHDA morphism, *i.e.* that $\pi_X \tilde{\delta}^\nu_k = \delta^\nu_k \pi_X$, is again trivial for $\nu = 1$ and more complicated for $\nu = 0$.

The proof that $\tilde{X}$ is a higher-dimensional tree is in appendix. If X itself is a higher-dimensional tree, then an inverse to π_X is given by mapping $x \in X$ to the unique homotopy class $[x_1, \dots, x_m] \in \tilde{X}$ of any pointed cube path $(x_1, \dots, x_m)$ in X with $x_m = x$. ◀

▶ **Theorem 15.** *Projections $\pi_X : \tilde{X} \to X$ are open, hence any PHDA is hd-bisimilar to its unfolding.*

Transitivity of hd-bisimilarity now implies the following, relating hd-bisimilarity of PHDA to hd-bisimilarity of homotopy classes of computations. This will be central in our comparison to other equivalences in Section 6.

▶ **Corollary 16.** *PHDA X, Y are hd-bisimilar iff their unfoldings $\tilde{X}, \tilde{Y}$ are hd-bisimilar.*

6 Relation to Other Equivalences

We now lift hd-bisimilarity to the labeled setting and relate it to other equivalences for concurrent models. We will show that hd-bisimilarity is implied by history-preserving bisimilarity, but not by split bisimilarity. As LHDA $= * \downarrow$ PPS $\downarrow !\Sigma$ is defined as a double

comma category, our notions of open maps and hd-bisimilarity trivially carry over; in LHDA, these are now required to preserve labels.

We recall the notions of history-preserving bisimilarity, ST-bisimilarity and split bisimilarity from [30] (and extend them to partial HDA). For a labeled PHDA $* \to X \xrightarrow{\lambda} !\Sigma$, we extend λ to cube paths in X by $\lambda(x_1, \dots, x_m) = (\lambda(x_1), \dots, \lambda(x_m))$. Note that there is a one-to-one correspondence between label sequences $\lambda(\rho)$ and *split traces*, see [30, Sect. 7.5]. Below we use $\rightsquigarrow$ for cube path extensions, *i.e.* $\rho \rightsquigarrow \rho'$ iff ρ is a prefix of ρ'.

Labeled PHDA $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$, $* \xrightarrow{j} Y \xrightarrow{\mu} !\Sigma$ are *split bisimilar* iff there exists a relation R between pointed cube paths in X and pointed cube paths in Y for which $((i),(j)) \in R$, and such that for all $(\rho, \sigma) \in R$,

(1) $\lambda(\rho) = \mu(\sigma)$,
(2) for all $\rho \rightsquigarrow \rho'$ there exists $\sigma \rightsquigarrow \sigma'$ with $(\rho', \sigma') \in R$, and
(3) for all $\sigma \rightsquigarrow \sigma'$ there exists $\rho \rightsquigarrow \rho'$ with $(\rho', \sigma') \in R$.

X and Y are *ST-bisimilar* if, instead of condition (1) above, it holds that

(1′) *ST-trace*$(\rho) =$ *ST-trace*(σ).

Here *ST-trace*(ρ) is the *ST-trace* of ρ defined by annotating *split-trace*(ρ) with a mapping which gives the starting point of any terminating action, see [30] (this is important for auto-concurrency). X and Y are *history-preserving bisimilar* iff (1′), (2) and (3) hold and, additionally, for all $(\rho, \sigma) \in R$ and all p,

(4) for all $\rho \overset{p}{\sim} \rho'$, there exists $\sigma \overset{p}{\sim} \sigma'$ with $(\rho', \sigma') \in R$, and
(5) for all $\sigma \overset{p}{\sim} \sigma'$, there exists $\rho \overset{p}{\sim} \rho'$ with $(\rho', \sigma') \in R$.

▶ **Example 11** (contd.)**.** In the example in Fig. 4 above, there is an ST-bisimilarity relation which relates the cube path $(x_0, y_1, x_1, y_3, x_3)$ to $(x'_0, y'_1, x'_1, y'_4, x'_4)$, and in fact any ST-bisimilarity needs to do so. But then $(x'_0, y'_1, x'_1, y'_4, x'_4)$ is 3-adjacent to $(x'_0, y'_1, z', y'_4, x'_4)$, whereas $(x_0, y_1, x_1, y_3, x_3)$ admits no 3-adjacency. Hence these HDA are ST-bisimilar but not history-preserving bisimilar.

The following theorem expresses hd-bisimilarity in a way comparable to the above definitions.

▶ **Theorem 17.** *Labeled PHDA* $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$, $* \xrightarrow{j} Y \xrightarrow{\mu} !\Sigma$ *are hd-bisimilar iff there exists a relation R between pointed cube paths in X and pointed cube paths in Y for which $((i),(j)) \in R$, and such that for all $(\rho, \sigma) \in R$,*

1. *$\lambda(\rho) \sim \mu(\sigma)$,*
2. *for all $\rho \rightsquigarrow \rho'$, there exists $\sigma \rightsquigarrow \sigma'$ with $(\rho', \sigma') \in R$,*
3. *for all $\sigma \rightsquigarrow \sigma'$, there exists $\rho \rightsquigarrow \rho'$ with $(\rho', \sigma') \in R$,*
4. *for all $\rho \sim \rho'$, there exists $\sigma \sim \sigma'$ with $(\rho', \sigma') \in R$, and*
5. *for all $\sigma \sim \sigma'$, there exists $\rho \sim \rho'$ with $(\rho', \sigma') \in R$.*

▶ **Example 11** (contd.)**.** Continuing the example in Fig. 4 above, a hd-bisimilarity relation as in Theorem 17 relates the cube path $(x_0, y_1, x_1, y_3, x_3)$ to $(x'_0, y'_1, x'_1, y'_4, x'_4)$, but also to $(x'_0, y'_1, z', y'_4, x'_4)$ and to any other cube path in X' homotopic to $(x'_0, y'_1, x'_1, y'_4, x'_4)$.

▶ **Theorem 18.** *Hd-bisimilarity is strictly weaker than history-preserving bisimilarity, but not weaker than split bisimilarity.*

Proof. When comparing the conditions in Theorem 17 with the ones for history-preserving bisimilarity above, we see that $\lambda(\rho) = \mu(\sigma)$ implies $\lambda(\rho) \sim \mu(\sigma)$ and adjacency implies homotopy. (For history-preserving bisimilarity, the adjacencies are required to happen *in the same place* in the cube paths.) Thus history-preserving bisimilarity implies hd-bisimilarity.

In Example 11 we have seen two labeled HDA which are hd-bisimilar but not history-preserving bisimilar, hence hd-bisimilarity is strictly weaker than history-preserving bisimilarity. Example 19 below will expose two labeled HDA which are split bisimilar but not hd-bisimilar, showing the last claim of the theorem. ◀

▶ **Example 19.** Using a hd-bisimulation game, we show that the HDA in Fig. 6 are not hd-bisimilar. Note that according to [34], they are split bisimilar. This shows that split bisimilarity does not imply hd-bisimilarity. From the initial configuration (x_0, x'_0) of the game, the spoiler plays y_1, to which the duplicator can only answer y'_1. Then the spoiler plays z_1, with only possible answer z'_1, leading to the configuration (z_1, z'_1). Playing y_4 and then z_2, the spoiler forces the configuration (z_2, z'_2) and, playing y_8 and then z_4, leads the game to the *cc*-labeled configuration (z_4, z'_4). Here the spoiler plays y_{12}, which the duplicator has to answer by the z'_4-boundary *in the same direction*, hence y'_{12}. But then the spoiler can play the *cd*-labeled z_5, to which the duplicator has no answer.

7 Conclusion and Further Work

We have introduced a generalization of higher-dimensional automata, partial HDA, which alleviates some modeling shortcomings of HDA. We have seen that PHDA are useful for modeling priorities and the disabling of events by other events, but they should also be useful for example in the context of left-merge (*e.g.* in ACP [2]) and other asymmetric operators.

We have seen that PHDA have a natural notion of (higher-dimensional) bisimilarity, which is polynomial-time decidable for finite PHDA. We have lifted this notion to a relation on computations in PHDA and seen that it is strictly weaker than history-preserving bisimilarity but not weaker than split bisimilarity, but its precise placement in the concurrent hierarchy, especially its relation with ST-bisimilarity, remains open.

To the best of our knowledge, hd-bisimilarity is the first useful equivalence notion for concurrent systems which is defined directly on the structure, instead of on computations. This is important from a practical point of view: we have seen that it can be decided using a simple fixed-point algorithm, or alternatively using a sort of higher-dimensional bisimulation game. We plan to implement these algorithms in a tool for equivalence checking of PHDA; this would make equivalence checking of concurrent systems feasible in practice.

References

1 Marek A. Bednarczyk. *Categories of asynchronous systems.* PhD thesis, University of Sussex, UK, 1987.

2 Jan A. Bergstra and Jan W. Klop. Process algebra for synchronous communication. *Inf. Cont.*, 60(1-3):109–137, 1984.

3 Ronald Brown and Philip J. Higgins. On the algebra of cubes. *J. Pure Appl. Alg.*, 21:233–260, 1981.

4 Uli Fahrenberg. A category of higher-dimensional automata. In *FOSSACS*, volume 3441 of *Lect. Notes Comput. Sci.*, pages 187–201. Springer, 2005.

5 Uli Fahrenberg. *Higher-Dimensional Automata from a Topological Viewpoint.* PhD thesis, Aalborg University, Denmark, 2005.

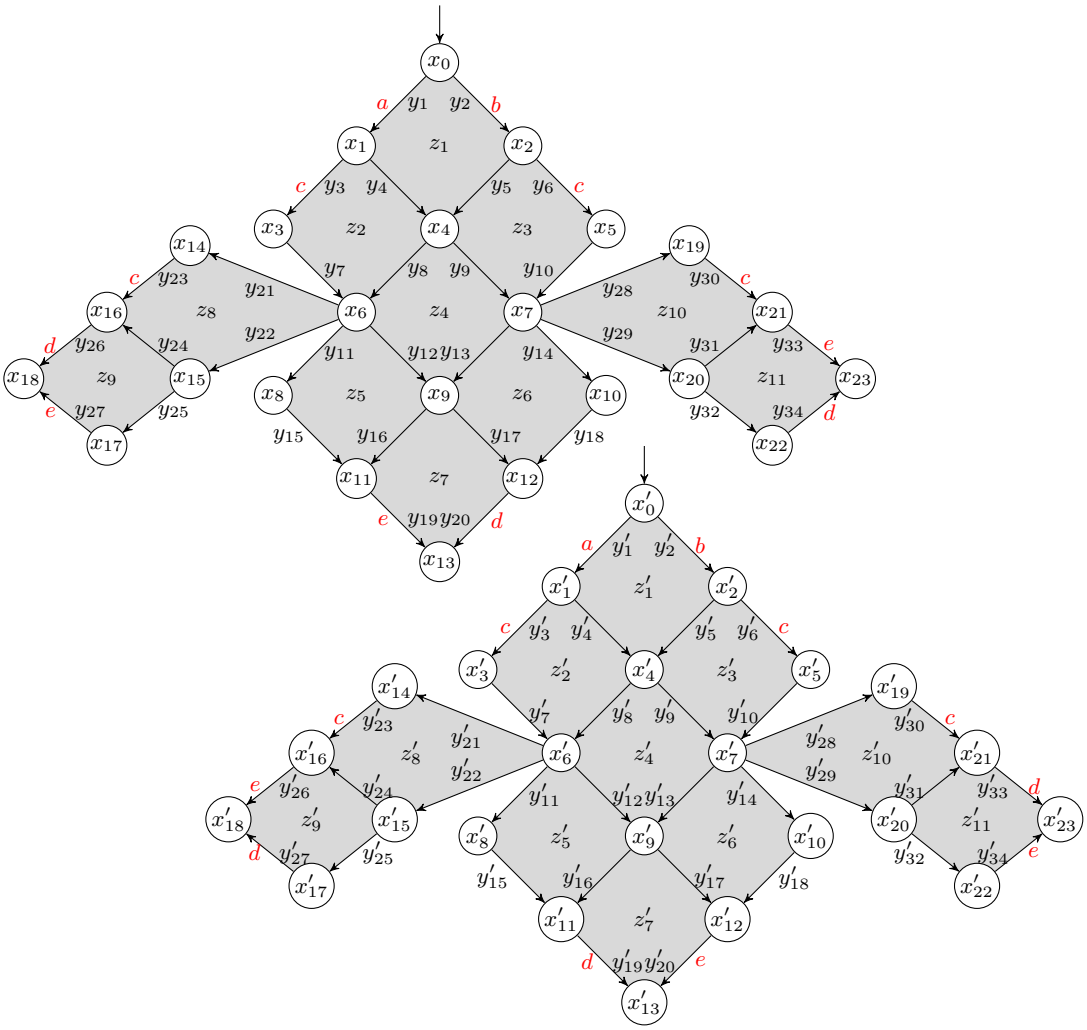


Figure 6 Two HDA pertaining to Example 19.

6 Uli Fahrenberg and Axel Legay. History-preserving bisimilarity for higher-dimensional automata via open maps. *Electr. Notes Theor. Comput. Sci.*, 298:165–178, 2013.

7 Uli Fahrenberg and Axel Legay. Homotopy bisimilarity for higher-dimensional automata. *CoRR*, abs/1409.5865, 2014.

8 Lisbeth Fajstrup. Dipaths and dihomotopies in a cubical complex. *Adv. Appl. Math.*, 35(2):188–206, 2005.

9 Lisbeth Fajstrup, Éric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raußen. Trace spaces: An efficient new technique for state-space reduction. In Helmut Seidl, editor, *ESOP*, volume 7211 of *Lecture Notes in Computer Science*, pages 274–294. Springer, 2012.

10 Lisbeth Fajstrup, Martin Raussen, and Éric Goubault. Algebraic topology and concurrency. *Theor. Comput. Sci.*, 357(1-3):241–278, 2006.

11 Philippe Gaucher. Homotopical interpretation of globular complex by multipointed d-space. *Theory Appl. Categories*, 22:588–621, 2009.

12 Philippe Gaucher. Towards a homotopy theory of higher dimensional transition systems. *Theory Appl. Categories*, 25:295–341, 2011.

13 Éric Goubault. Geometry and concurrency: A user's guide. *Math. Struct. Comput. Sci.*, 10(4):411–425, 2000.

14 Éric Goubault and Thomas P. Jensen. Homology of higher dimensional automata. In Rance Cleaveland, editor, *CONCUR*, volume 630 of *Lect. Notes Comput. Sci.*, pages 254–268. Springer, 1992.

15 Éric Goubault and Samuel Mimram. Formal relationships between geometrical and classical models for concurrency. *Electronic Notes in Theoretical Computer Science*, 283:77–109, 2012.

16 Marco Grandis. *Directed algebraic topology: models of non-reversible worlds*. New mathematical monographs. Cambridge Univ. Press, 2009.

17 André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Inf. Comput.*, 127(2):164–185, 1996.

18 Dexter Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer, 1997.

19 Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

20 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.*, 13:85–108, 1981.

21 David M.R. Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *IFIP TCS*, volume 104 of *Lect. Notes Comput. Sci.*, pages 167–183. Springer, 1981.

22 Carl A. Petri. *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

23 Vaughan Pratt. Modeling concurrency with geometry. In *POPL*, pages 311–322. ACM Press, 1991.

24 Cristian Prisacariu. The glory of the past and geometrical concurrency. In Andrei Voronkov, editor, *Turing-100*, volume 10 of *EPiC*, pages 252–267. EasyChair, 2012.

25 Alexander M. Rabinovich and Boris A. Trakhtenbrot. Behavior structures and nets. *Fund. Inf.*, 11(4):357–403, 1988.

26 Jean-Pierre Serre. *Homologie singulière des espaces fibrés*. PhD thesis, Ecole Normale Supérieure, 1951.

27 Mike W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

28 Colin Stirling. Modal and temporal logics for processes. In *Proc. Banff Higher Order Workshop*, volume 1043 of *Lect. Notes Comput. Sci.*, pages 149–237. Springer, 1995.

29 Rob J. van Glabbeek. Bisimulations for higher dimensional automata. Email message, June 1991. `http://theory.stanford.edu/~rvg/hda`.

30 Rob J. van Glabbeek. On the expressiveness of higher dimensional automata. *Theor. Comput. Sci.*, 356(3):265–290, 2006.

31 Rob J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Inf.*, 37(4/5):229–327, 2001.

32 Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *Theor. Comput. Sci.*, 410(41):4111–4159, 2009.

33 Rob J. van Glabbeek and Frits W. Vaandrager. Petri net models for algebraic theories of concurrency. In J. W. de Bakker, A. J. Nijman, and Philip C. Treleaven, editors, *PARLE (2)*, volume 259 of *Lect. Notes Comput. Sci.*, pages 224–242. Springer, 1987.

34 Rob J. van Glabbeek and Frits W. Vaandrager. The difference between splitting in n and n+1. *Inf. Comput.*, 136(2):109–142, 1997.

35 Glynn Winskel and Mogens Nielsen. Models for concurrency. In Samson Abramsky, Dov M. Gabbay, and Thomas S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4. Clarendon Press, Oxford, 1995.